Connection.c

I connection filen føler jeg at jeg har ganske god kontroll på hva jeg gjør på noen av funksjonene, men jeg er mer usikre på andre. I tcp_connect bruker jeg en sockadd_in struct for å lagre ip adressen og portnummeret. På denne måten kan jeg enkelt bruke connect funksjonen til å koble opp destinasjonsadressen med TCP socketen.

En annen viktig funksjon er tcp_create_and_listen som returnerer en socket som jeg kaller moteplass. Dette er fordi denne blir senere blir gjort om til en passiv socket som venter på koblinger fra andre koblinger i accept i tcp_accept. Jeg har gjort backlogen til denne til 26 fordi det er 26 mulige connections på samme tid hvis hver ID er en bokstav fra A-Z. Tcp_create_and_listen og tcp_accept er de to viktigste funksjonene i main loopen i proxy.c for at server socketen skal oppdage nye connections.

Proxy.c

I main funksjonen til proxy starter jeg med å lage en server socket for serveren som lytter til nye koblinger med tcp_create_and_listen. Jeg lager deretter et fd_set som innehlder en mengde med file descriptors og deres tilhørende sockets, jeg initialiserer så denne megnden med FD_ZERO og legger først til server socketen.

I main looper starter jeg hver iterasjon med å lage en kopi av mengden file descriptors i en variabel kaldt read_fds. Dette er for å bevare orginal mengden med file desciprtors da select funksjonen modifiserer mengden den får inn som argument for å indikere hvilken file desciriptor det er oppdaget aktivitet. Etter select bruker jeg FD_ISSET for å sjekke spesifikt hvilken socket det har skjedd noen aktivitet. Hvis det er server socketen betyr det at det har inkommet en ny kobling på server socketen, med andre ord har en ny client koblet til. Hvis det er server socketen det har vært aktivitet på kaller jeg på handleNewClient(). I denne funksjonen setter jeg lager jeg plass til ny Client structs og setter variable verdier tilhørende structen som ID og hvorvidt Clienten som har koblet seg på sender data i binært format eller ikke. Her tildeler også tcp_accept nye sockets til nylig tilkoblede Clients som vil bli brukt for videre kommunikasjon til serveren. Clienten blir også lagt til en en array av type Clients. Denne arrayen har en maks størrelse på 26 igjen fordi dette er antallet IDer som finnes mellom A-Z. Jeg antar at ikke flere connections med ID A kan være koblet på samtidig. Tilbake til main loopen vil den nye Clienten sin socket bli lagt til i mengden med sockets som select hører etter.

Etter main loopen har sjekker om det er aktivitet på server socketen vil den loope gjennom alle clients i Client arrayet og sjekke hver Client sin socket for aktivitet, igjen med FD_ISSET. Hver Client struct har en socket som variabel slik at jeg enkelt kan sjekke alle Clients som allerede er tilkoblet. Hvis det er noen aktivitet på en Client socket, sender jeg denne Clienten som argument til handleClient. I handleClient er det viktigste som skjer at dataen i enten XML format eller binært format blir oversatt til en Record struct. Jeg har valgt å lage denne structen fordi dette gjøre det enklere å oversette mellom ulike formater gitt hjelpefunksjonene i precoden. Jeg sjeker først om ingen bytes er lest, som enten kan bety at socketen har blirr lukket av den andre siden eller at det ikke finnes noen flere bytes å lese. Uansett kaller jeg på removeClient i dette tilfellet, der intensjonen er at jeg skal rydde opp datastrukturer som ikke lenger skal bli brukt. Deretter sjekker jeg om Clienten sender i binært format eller ikke, og kaller på XMLtoRecord eller BinarytoRecord etterom hva som er

tilfellet. Videre kaller handleClient på forward message. Her antar jeg at hvis handleClient blir kallt på, betyr det at det er en Client som skal sende data. I forward message looper jeg gjennom alle Clients som er koblet til serveren og kaller på recordToXML eller recordToBinary og skriver til riktig receiver i riktig format.

RecordFromFormat.c

I denne filen har jeg laget en funksjon som heter get_attribute_value. På dennen måten kan jeg kalle samme funksjon flere ganger for å finne de verdiene jeg har lyst på i en XML fil, så slipper jeg å skrive samme kode flere ganger. Attribute value returnerer NULL hvis verdien den leter etter ikke finne i en XML fil, på denne måten kaller jeg bare på funksjonene fra record.c verdien jeg leter etter finnes i XML filen. På denne måten vil det ikke oppstå problemer hvis det skulle sendes en XML fil med manglende verdier. Til slutt sjekker jeg om en dest er funnet i XML filen, etter som dette er kravet for at en Record skal være fullstendig. I BinaryToRecord funksjonen definerer jeg en slags "peker" som holder kontroll på hvor mange bytes jeg har lest fra en fil. På denne måten kan jeg sende denne pekeren som verdi nåe jeg ønsker å finne spesifike data i en binær fil. Jeg bruker ntohl når jeg finner lengden på brukernavnet fordi denne kommer i nettverk bytes rekkefølge.